CS8603 - DISTRIBUTED SYSTEMS

CS8603 DISTRIBUTED SYSTEMS L T P C
3 0 0 3

OBJECTIVES:

- To understand the foundations of distributed systems.
- To learn issues related to clock Synchronization and the need for global state in distributed systems.
- To learn distributed mutual exclusion and deadlock detection algorithms.
- To understand the significance of agreement, fault tolerance and recovery protocols in Distributed Systems.
- To learn the characteristics of peer-to-peer and distributed shared memory systems.

UNIT I INTRODUCTION 9

Introduction: Definition –Relation to computer system components –Motivation –Relation to parallel systems – Message-passing systems versus shared memory systems –Primitives for distributed communication –Synchronous versus asynchronous executions –Design issues and challenges. **A model of distributed computations**: A distributed program –A model of distributed executions –Models of communication networks –Global state – Cuts –Past and future cones of an event –Models of process communications. **Logical Time:** A framework for a system of logical clocks –Scalar time –Vector time – Physical clock synchronization: NTP.

UNIT II MESSAGE ORDERING & SNAPSHOTS 9

Message ordering and group communication: Message ordering paradigms –Asynchronous execution with synchronous communication –Synchronous program order on an asynchronous system –Group communication – Causal order (CO) - Total order. **Global state and snapshot recording algorithms:** Introduction –System model and definitions –Snapshot algorithms for FIFO channels

UNIT III DISTRIBUTED MUTEX & DEADLOCK 9

Distributed mutual exclusion algorithms: Introduction – Preliminaries – Lamport‘s algorithm – Ricart-Agrawala algorithm – Maekawa‘s algorithm – Suzuki–Kasami‘s broadcast algorithm. **Deadlock detection in distributed systems:** Introduction – System model – Preliminaries – Models of deadlocks – Knapp‘s classification – Algorithms for the single resource model, the AND model and the OR model.

UNIT IV RECOVERY & CONSENSUS 9

Checkpointing and rollback recovery: Introduction – Background and definitions – Issues in failure recovery – Checkpoint-based recovery – Log-based rollback recovery – Coordinated checkpointing algorithm – Algorithm for asynchronous checkpointing and recovery. **Consensus and agreement algorithms:** Problem definition – Overview of results – Agreement in a failure – free system – Agreement in synchronous systems with failures.

UNIT V P2P & DISTRIBUTED SHARED MEMORY 9

Peer-to-peer computing and overlay graphs: Introduction – Data indexing and overlays – Chord – Content addressable networks – Tapestry. **Distributed shared memory:** Abstraction and advantages – Memory consistency models –Shared memory Mutual Exclusion.

TOTAL: 45 PERIODS

OUTCOMES:

At the end of this course, the students will be able to:

- Elucidate the foundations and issues of distributed systems
- Understand the various synchronization issues and global state for distributed systems.
- Understand the Mutual Exclusion and Deadlock detection algorithms in distributed systems
- Describe the agreement protocols and fault tolerance mechanisms in distributed systems.
- Describe the features of peer-to-peer and distributed shared memory systems

TEXT BOOKS:

1. Kshemkalyani, Ajay D., and Mukesh Singhal. Distributed computing: principles, algorithms, and systems. Cambridge University Press, 2011.
2. George Coulouris, Jean Dollimore and Tim Kindberg, ―Distributed Systems Concepts and Design‖, Fifth Edition, Pearson Education, 2012.

REFERENCES:

1. Pradeep K Sinha, "Distributed Operating Systems: Concepts and Design", Prentice Hall of India, 2007. 2. Mukesh Singhal and Niranjan G. Shivaratri. Advanced concepts in operating systems. McGraw-Hill, Inc., 1994.
3. Tanenbaum A.S., Van Steen M., ―Distributed Systems: Principles and Paradigms‖, Pearson Education, 2007.
4. Liu M.L., ―Distributed Computing, Principles and Applications‖, Pearson Education, 2004.
5. Nancy A Lynch, ―Distributed Algorithms‖, Morgan Kaufman Publishers, USA, 2003.

COURSE OUTCOME

At the end of course, students will have an

CO1	Understand the basics and challenges of distributed system
CO2	Understand the various synchronization issues and global state for distributed systems.
CO3	Understand the Mutual Exclusion and Deadlock detection algorithms in distributed systems
CO4	Analyze the recovery & consensus mechanisms in distributed systems
CO5	Understand the features of peer-to-peer and distributed shared memory systems
CO6	Understand the overall advancement in computing using Distributed Systems.

UNIT I INTRODUCTION

Introduction: Definition –Relation to computer system components –Motivation –Relation to parallel systems – Message-passing systems versus shared memory systems –Primitives for distributed communication –Synchronous versus asynchronous executions –Design issues and challenges. A model of distributed computations: A distributed program –A model of distributed executions –Models of communication networks –Global state – Cuts – Past and future cones of an event –Models of process communications. Logical Time: A framework for a system of logical clocks –Scalar time –Vector time – Physical clock synchronization: NTP.

1. **Define Distributed systems. (R)**

   A distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved.

2. **Discuss the features of distributed systems (U)**
   - No common physical clock
   - Geographical separation
   - No shared memory
   - Autonomy and heterogeneity.

3. **Define distributed execution (R)**

   A distributed execution is the execution of processes across the distributed system to collaboratively achieve a common goal. An execution is also sometimes termed a computation or a run.

4. **Mention the motivation factors of distributed systems.(U)**
   - Inherently distributed computations
   - Resource sharing
   - Access to geographically remote data and resources
   - Enhanced reliability
   - Increased performance/cost ratio
   - Scalability
   - Modularity and incremental expandability

5. **State the characteristics of parallel systems (U)**

   A multiprocessor system is a parallel system in which the multiple processors have direct access to shared memory which forms a common address space.

   A multicomputer parallel system is a parallel system in which the multiple processors do not have direct access to shared memory.

   Array processors belong to a class of parallel computers that are physically co-located, are very tightly coupled, and have a common system clock

6. State flynn's taxonomy (U)

Flynn's taxonomy is a specific classification of parallel computer architectures that are based on the number of concurrent instruction (single or multiple) and data streams (single or multiple) available in the architecture.

7. Mention the four categories in Flynn's taxonomy.(U)

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data stream (SIMD)
- Multiple instruction stream, single data stream (MISD)
- Multiple instruction stream, multiple data stream (MIMD**)**

8. Define coupling.(R)

The degree of coupling among a set of modules, whether hardware or software ,is measured in terms of the interdependency and binding and/or homogeneity among the modules.

9. Differentiate Tightly coupled and Loosely coupled multiprocessors (U)

Tightly coupled	**Loosely coupled**
Switch based or Bus based	Bus Based or using a more general communication network and the processors may be heterogeneous
Tightly coupled multiprocessors are NUMA Share memory or that communicate by message passing	Loosely coupled multicomputer are without shared memory and without common clock that are physically remote.

10. Define concurrency of a program (R)

The parallelism/concurrency in a parallel/distributed program can be measured by the ratio of the number of local (non-communication and non-shared memory access) operations to the total number of operations, including the communication or shared memory access operations.

11. Define granularity of a program (R)

The ratio of the amount of computation to the amount of communication within the parallel/distributed program is termed as granularity.

12. Mention the primitives for distributed communication (U)

Blocking/non-blocking, synchronous/asynchronous primitives
Processor synchrony
Libraries and standards

13. Differentiate synchronous and asynchronous executions.(U)

Synchronous Execution	**Asynchronous Execution**
Processors are synchronised and the clock drift rate between any two processors is bounded	No processor synchrony and there is no bound on the drift rate of processor clocks
Message delivery times occur in one logical step or round	Message delays are finite but unbounded

Known upper bound on a time taken by a process to execute a step	No upper bound on the time taken by a process to execute a step

14. State the design issues and challenges.(U) April/May 2021

- Communication
- Processes
- Naming
- Synchronization
- Data storage and access
- Consistency and replication
- Fault tolerance
- Security
- Applications Programming Interface (API) and transparency

15. Define event streaming.(R)

An important paradigm for monitoring distributed events is that of event streaming, wherein streams of relevant events reported from different processes are examined collectively to detect predicates.

16. Discuss the various strategies of a reliable and fault –tolerant distributed systems.(U)

- Consensus algorithm
- Replication and replica management
- Voting and quorum systems
- Distributed databases and distributed commit
- Self-stabilizing systems
- Check pointing and recovery algorithms
- Failure detectors

17. Define load balancing and mention its forms (R)

Load balancing may be necessary because of a variety of factors such as high network traffic or high request rate causing the network connection to be a bottleneck, or high computational load.

Forms of load balancing

- Data migration
- Computation migration
- Distributed scheduling

18. State the applications of distributed computing.(R)

- Mobile systems
- Sensor networks
- Ubiquitous or pervasive computing
- Peer-to-peer computing
- Publish-subscribe, content distribution, and multimedia

- Distributed agents
- Distributed data mining
- Grid computing
- Security in distributed systems

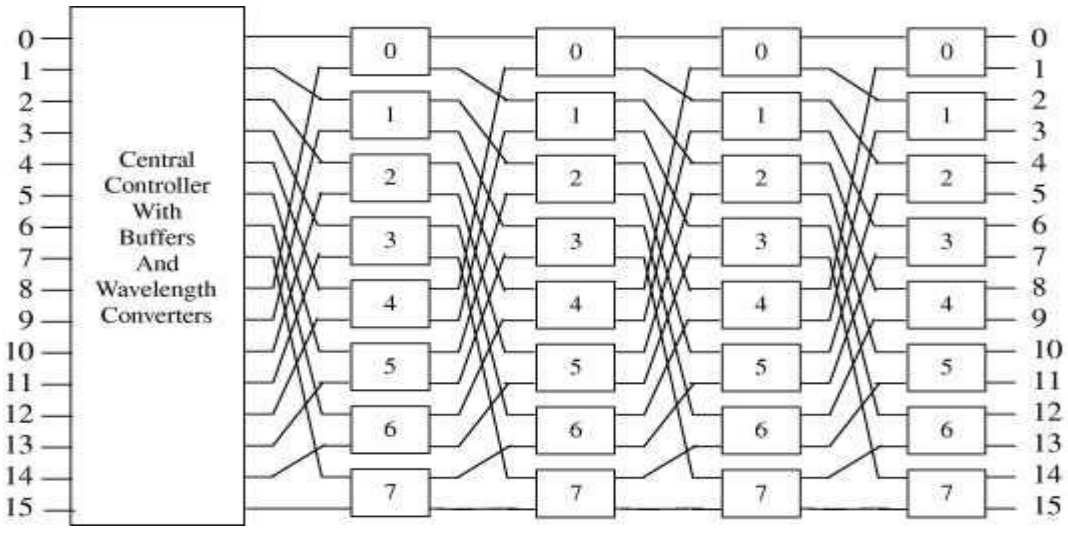
19. What are the main differences between a parallel system and a distributed system? (U)

Parallel Computing	**Distributed Computing**
Type of Computation in which many calculations or the execution of processes are carried out simultaneously	A system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another
Occurs in a single computer	Involves multiple computers
Multiple processors execute multiple tasks at the same time	Multiple computers performs tasks at the same time
Computer can have shared memory or distributed memory	Each computer has its own memory
Processors communicate with each other using a bus	Computers communicate with each other via the network
Increase the performance of the systems	Allows scalability, sharing resources and helps to perform computation tasks efficiently

20. Explain why a Receive call cannot be asynchronous.(U)

Asynchronous message passing allows more parallelism. Since a process does not block, it can do some computation while the message is in transit. In the case of receive; this means a process can express its interest in receiving messages on multiple ports simultaneously.

21. Draw the Omega and Butterfly networks for n = 16 inputs and outputs.(U)



Block diagram for a 16×16 omega network structure with central controller and central buffer.

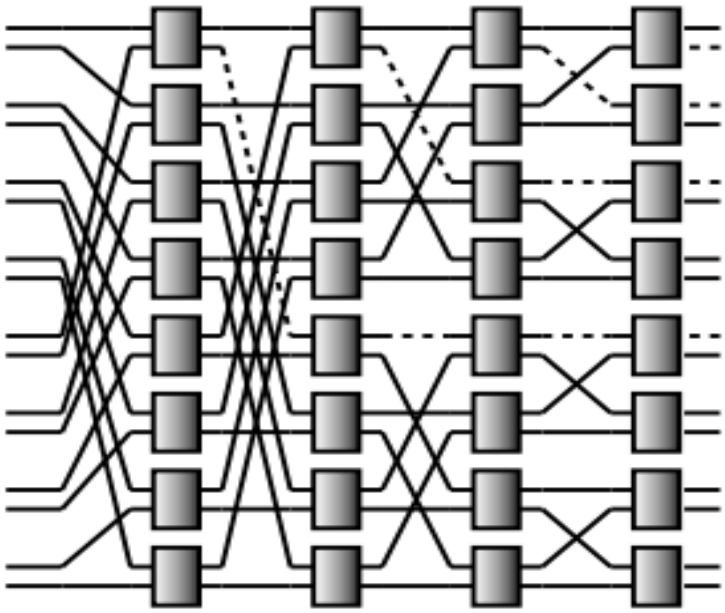


Fig : 16 * 16 Butterfly network structure

22. Write down the principles of distributed systems. (U)

- Communication
- Coordination
- Fault tolerance
- Locality
- Parallelism

23. Give examples of distributed systems. (U)

- Intranet and Local Area Network
- Internet/World-Wide Web
- Mobile and Ubiquitous Computing
- Database Management System
- Automatic Teller Machine Network
- E-mail
- Newsgroups and bulletin boards
- FTP, Telnet
- W WW
- Distributed computing
- Electronic payment
- Distributed real-time processing

24. What is Logical and physical Concurrency.(R)

In a distributed computation, two events are logically concurrent if and only if they do not causally affect each other. Physical concurrency, on the other hand, has a connotation that the events occur at the same instant in physical time.

25. Mention the various models of communication networks.(U)

1. **FIFO (first-in, first-out)-** each channel acts as a first-in first-out message queue and thus,message ordering is preserved by a channel.
2. **Non-FIFO** - a channel acts like a set in which the sender process adds messages and the receiver process removes messages from it in a random order
3. **Causal ordering**- based on Lamport's "happens before" relation.

26. Define global state .(R)

The global state of a distributed system is a collection of the local states of its components, namely, the processes and the communication channels. the global state GS is defined as

$$GS = \{\bigcup_i LS_i^{x_i}, \bigcup_{j,k} SC_{jk}^{y_j - z_k}\}.$$

27. Define consistent global state.(R)

A message cannot be received if it was not sent; that is, the state should not violate causality. Such states are called consistent global states and are meaningful global states.

28. What is a cut ?(R)

In the space–time diagram of a distributed computation, a zigzag line joining one arbitrary point on each process line is termed a cut in the computation. Such a line slices the space–time diagram, and thus the set of events in the distributed computation, into a PAST and a FUTURE.

29. Illustrate past and future cones of an event.(U)

In a distributed computation, an event ej could have been affected only by all events ei such that e i →ej and all the information available at ei could be made accessible at ej . All such events ei belong to the past of ej.

The future of an event ej denoted by Future(e j), contains all events ei that are causally affected by ej

30. State the models of process communication.(U)

There are two basic models of process communications such as synchronous and asynchronous.

The synchronous communication model is a blocking type where on a message send, the sender process blocks until the message has been received by the receiver process.

The asynchronous communication model is a non-blocking type where the sender and the receiver do not synchronize to exchange a message.

31. Define logical clock.(R)

The logical clock C is a function that maps an event e in a distributed system to an element in the time domain T, denoted as C(e) and called the timestamp of e, and is defined as follows: C : H _ → T,

32. Define scalar time.(R)

The scalar time representation was proposed by Lamport in 1978 as an attempt to totally order events in a distributed system. Time domain in this representation is the set of non-negative integers.

33. Mention the properties of scalar time.(U)

Consistency property
Total Ordering
Event counting
No strong consistency

34. Define vector time.(R)

The system of vector clocks was developed independently by Fidge, Mattern, and Schmuck. In the system of vector clocks, the time domain is represented by a set of n-dimensional non-negative integer vectors.

35. Mention the properties of vector time.(U)

Isomorphism
Strong consistency
Event counting

36. Define offset.(R)

Offset Clock offset is the difference between the time reported by a clock and the *real time*. The offset of the clock C_a is given by $C_a(t) - t$. The offset of clock C_a relative to C_b at time $t \geq 0$ is given by $C_a(t) - C_b(t)$.

37. Define Skew (R)

Skew The skew of a clock is the difference in the frequencies of the clock and the perfect clock. The skew of a clock C_a relative to clock C_b at time t is $C_a'(t) - C_b'(t)$.
If the skew is bounded by ρ, then as per Eq.(3.1), clock values are allowed to diverge at a rate in the range of $1 - \rho$ to $1 + \rho$.

38. Define drift rate.(R)

Drift (rate) The drift of clock C_a is the second derivative of the clock value with respect to time, namely, $C_a''(t)$. The drift of clock C_a relative to clock C_b at time t is $C_a''(t) - C_b''(t)$.

39. Define Network Time Protocol(NTP) (R)

The Network Time Protocol (NTP), which is widely used for clock synchronization on the Internet, uses the the offset delay estimation method.

40. Why do we need a distributed system (U) April/May 2021

Distributed systems provide scalability and improved performance in ways that monolithic systems can't, and because they can draw on the capabilities of other computing devices and processes, distributed systems can offer features that would be difficult or impossible to develop on a single system.

Part – A

1. Why do we need a distributed system? **April/May 2021**
2. List out the distributed system challenges. **April/May 2021**
3. Name the primitives for distributed communication. **Nov/Dec 2021**
4. Compare message passing systems and shared memory systems. **Nov/Dec 2021**
5. What do you mean by message passing? **Nov/Dec 2022**
6. Define: Distributed program **Nov/Dec 2022**

Part - B

1. Why global states are essential in distributed computing systems? Elaborate with an example. **Nov/Dec 2021**
2. Elaborate any two logical dock categories in distributed systems with an example. **Nov/Dec 2021**
3. How do you classify a parallel system and brief them? **April/May 2021**
4. Compare Synchronous versus asynchronous execution. **April/May 2021**
5. What are the functions must be addressed while designing and building a distributed system ? Explain. **April/May 2021**
6. Illustrate difference between message passing and shared memory process communication model**. Nov/Dec 2022**
7. Explain the types of group communication used in distributed system. **Nov/Dec 2022**
8. Elaborate the design issues and challenges in building distributed system.
9. Explain the characteristics of parallel systems with examples.(U)
10. Discuss Flynn's taxonomy. (U)
11. Compare message passing systems VS shared memory systems. (U)
12. Explain the primitives of distributed communication. (U)
13. List the various design issues and challenges of distributed systems. (U)
14. Categorize the application areas of distributed computing and newer challenges. (U)
15. Explain a model of distributed executions. (U)
16. Discuss the global state of a distributed system. (U)
17. Write notes on consistent cuts of a distributed computation. (U)
18. Explain the past and future cones of an event. (U)
19. Write notes on models of process communications. (U)
20. How to implement logical clock. (U)
21. Explain vector time with its properties and implementations. (U)
22. Discuss the physical clock synchronization using NTP.(U)
23. Compare Synchronous versus asynchronous execution(U)

UNIT II MESSAGE ORDERING & SNAPSHOTS

Message ordering and group communication: Message ordering paradigms –Asynchronous execution with synchronous communication –Synchronous program order on an asynchronous system –Group communication – Causal order (CO) - Total order. Global state and snapshot recording algorithms: Introduction –System model and definitions – Snapshot algorithms for FIFO channels

PART – A

1. **List out the ordering on messages. (U)**
   (i) non-FIFO,
   (ii) FIFO
   (iii) causal order
   (iv) Synchronous order

2. **Define asynchronous executions .(R)**
   An asynchronous execution (or A-execution) is an execution _E_ ≺_ for which the causality relation is a partial order.

3. **Define FIFO executions.(R)**
   Definition 6.2 (FIFO executions) A FIFO execution is an A-execution in which,
   for all (s, r) and $(s', r') \in \mathcal{T}$, $(s \sim s'$ and $r \sim r'$ and $s \prec s') \Longrightarrow r \prec r'$.

4. **Define casual order (CO).(R) April/May 2021**
   Definition 6.3 (Causal order (CO)) A CO execution is an A-execution in which,
   for all (s, r) and $(s', r') \in \mathcal{T}$, $(r \sim r'$ and $s \prec s') \Longrightarrow r \prec r'$.

5. **Define Message order (MO)(R)**
   Definition 6.5 (Message order (MO)) A MO execution is an A-execution in which,
   for all (s, r) and $(s', r') \in \mathcal{T}$, $s \prec s' \Longrightarrow \neg(r' \prec r)$.

6. **Define Synchronous execution.(R)**
   Definition 6.8 (Synchronous execution) A synchronous execution (or S-execution) is an execution $(E, \ll)$ for which the causality relation $\ll$ is a partial order.

7. **Differentiate synchronous with asynchronous execution.(R)**
   Synchronous Execution: All tasks within a block of code are all **executed** at the same time. **Asynchronous Execution**: All tasks within a block of code are not all **executed** at the same time.

8. **Define binary rendezevous and multiway rendezevous.(R)**
   Rendezvous between a pair of processes at a time, which is called binary rendezvous

One form of group communication is called multiway rendezvous, which is a synchronous communication among an arbitrary number of asynchronous processes.

9. Define group communication.(R)

A group is an operating system abstraction for a collective of related processes. The term multicast means the use of a single communication primitive to send a message to a specific set of processes rather than using a collection of individual point to point message primitives.

10. Define Crown.(R)

Definition 6.12 (Crown) Let E be an execution. A crown of size k in E is a sequence $\langle (s^i, r^i), i \in \{0, \ldots, k-1\} \rangle$ of pairs of corresponding send and receive events such that: $s^0 \prec r^1, s^1 \prec r^2, \ldots, s^{k-2} \prec r^{k-1}, s^{k-1} \prec r^0$.

11. Define multiway rendezevous.(R)

One form of group communication is called multiway rendezvous, which is a synchronous communication among an arbitrary number of asynchronous processes. All the processes involved "meet with each other," i.e., communicate "synchronously" with each other at one time.

12. Define Binary rendezvous.(R)

Rendezvous between a pair of processes at a time, which is called binary **Rendezvous** identifies **binary** code using a statistical model comprising instruction mnemonics, control flow sub-graphs and data constants which are simple to extract from a disassembly, yet normalizing with respect to different compilers and optimizations.

13. Define Implicit and explicit tracking.(R)

Explicit tracking Tracking of (source, timestamp, destination) information for messages (i) not known to be delivered and (ii) not guaranteed to be delivered in CO, is done explicitly using the l.

Implicit tracking Tracking of messages that are either (i) already delivered,or (ii) guaranteed to be delivered in CO, is performed implicitly.

14. Define Total order. (R)

For each pair of processes Pi and Pj and for each pair of messages Mx and My that are delivered to both the processes,Pi is delivered Mx before My if and only if Pj is delivered Mx before My.

15. State the centralized algorithm for total order.(U)

(1) When process P_i wants to multicast a message M to group G:
(1a) **send** $M(i, G)$ to central coordinator.

(2) When $M(i, G)$ arrives from P_i at the central coordinator:
(2a) **send** $M(i, G)$ to all members of the group G.

(3) When $M(i, G)$ arrives at P_j from the central coordinator:
(3a) **deliver** $M(i, G)$ to the application.

16. State three phases of distributed algorithm from the viewpoint of sender. (U)

Phase 1 In the first phase, a process multicasts the message M with a locally unique tag and the local timestamp to the group members.
Phase 2 In the second phase, the sender process awaits a reply from all the group members who respond with a tentative proposal for a revised timestamp for that message M. The **await** call is non-blocking,
Phase 3 In the third phase, the process multicasts the final timestamp to the group

17. State three phases of distributed algorithm from the viewpoint of receiver.(U)

Phase 1 In the first phase, the receiver receives the message with a tentative/proposed timestamp.
Phase 2 In the second phase, the receiver sends the revised timestamp back to the sender
Phase 3 In the third phase, the final timestamp is received from the multicaster

18. Define System model.(R)

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.

19. What is a consistent global state.(R)

A global state GS is a *consistent global state* iff it satisfies the following two conditions [16]:

C1: $send(m_{ij}) \in LS_i \Rightarrow m_{ij} \in SC_{ij} \oplus rec(m_{ij}) \in LS_j$ ($\oplus$ is the Ex-OR operator).
C2: $send(m_{ij}) \notin LS_i \Rightarrow m_{ij} \notin SC_{ij} \wedge rec(m_{ij}) \notin LS_j$.

20. What is a cut.(R)

A cut is a line joining an arbitrary point on each process line that slices the space–time diagram into a PAST and a FUTURE.

21. Define consistent cut.(R)

A consistent global state corresponds to a cut in which every message received in the PAST of the cut has been sent in the PAST of that cut. Such a cut is known as a consistent cut.

22. Mention the issues in recording a global state.(U)

I1: How to distinguish between the messages to be recorded in the snapshot (either in a channel state or a process state) from those not to be recorded.
The answer to this comes from conditions **C1** and **C2** as follows:
Any message that is sent by a process before recording its snapshot, must be recorded in the global snapshot (from **C1**).
Any message that is sent by a process after recording its snapshot, must not be recorded in the global snapshot (from **C2**).
I2: How to determine the instant when a process takes its snapshot. The answer to this comes from condition C2 as follows:
A process pj must record its snapshot before processing a message mij that was sent by process pi after recording its snapshot.

23. Define rubber-band criteria.(R)

Consider the two instants of recording of the local states in the banking example. If the cut formed by these instants is viewed as being an elastic band and if the elastic band is stretched so that it is vertical, then recorded states of all processes occur simultaneously at one physical instant, and the recorded global state occurs in the execution that is depicted in this modified space– time diagram. This is called the rubber-band criterion.

Part - A

1. Name the various message ordering paradigms used in distributed systems. **April/May 2021**
2. Define causal order execution. **April/May 2021**
3. Write simple example for message ordering. **Nov./Dec. 2021**
4. Identify the purpose of snapshot recording algorithm. **Nov./Dec. 2021**
5. What do you mean by synchronous and asynchronous execution? **Nov/Dec 2022**
6. What is meant by asynchronous programming? **Nov/Dec 2022**

Part - B

1. Discuss the purpose of message ordering paradigms and provide example for asynchronous execution communication in detail. **April/May 2021**
2. Describe the snapshot algorithms which could be applied for FIFO channels with diagrammatic representation. **April/May 2021**
3. Illustrate the necessary and sufficient conditions for causal ordering. **Nov./Dec. 2021**
4. Discuss in detail about Snapshot algorithms for FIFO channels. **Nov./Dec. 2021**
5. What are the four different types of ordering the messages? Explain. **Nov/Dec 2022**
6. Elucidate on the total and causal order in distributed system with neat diagram. **Nov/Dec 2022**
7. Design the procedure for causality in synchronous execution with a suitable example. **Nov/Dec 2022**
8. Illustrate the sufficient conditions causally ordered (CO) executions. (U)
9. Describe synchronous execution.(U)
10. Explain asynchronous execution with synchronous communication.(U)
11. Explain synchronous program order on an asynchronous system.(U)
12. Explain group communication.(U)
13. Explain the Raynal-Schiper –Toueg algorithm for casual order .(U)
14. Explain three phase distributed algorithm for total order.(U.
15. Explain System models in distributed environment.(U)
16. Write notes on (U)
    a. Consistent global state
    b. Interpretation in terms of cuts.
17. Explain Chandy and Lamport algorithm to record the global snapshot.(U)
18. Discuss in detail about the snapshot algorithm in FIFO Channels(U)
19. Mention the properties of the recorded global state.(U)

UNIT III DISTRIBUTED MUTEX & DEADLOCK

Distributed mutual exclusion algorithms: Introduction – Preliminaries – Lamport's algorithm – Ricart-Agrawala algorithm – Maekawa's algorithm – Suzuki–Kasami's broadcast algorithm. Deadlock detection in distributed systems: Introduction – System model – Preliminaries – Models of deadlocks – Knapp's classification – Algorithms for the single resource model, the AND model and the OR mode

PART A

1. **Mention the three basic approaches for implementing distributed mutual exclusion.(U)**
   a. Token-based approach
   b. Non-token-based approach
   c. Quorum-based approach.

2. **List the requirements of mutual exclusion algorithms.(U)**
   - Safety property
   - Liveness property
   - Fairness

3. **Mention the performance metrics of mutual exclusion.(U)**
   - Message complexity
   - Synchronization delay
   - Response time
   - System throughput

4. **List the deadlock handling messages.(U)**
   FAILED
   INQUIRE
   YIELD

5. **State Lamport algorithm.(R)**
   In Lamport's Algorithm critical section requests are executed in the increasing order of timestamps i.e a request with smaller timestamp will be given permission to execute critical section first than a request with larger timestamp. Three type of messages (**REQUEST**, **REPLY** and **RELEASE**) are used and communication channels are assumed to follow FIFO order.

6. **List the two types of messages used in Ricart - Agarwala algorithm**.(U)

   1. REQUEST

2. REPLY.

A process sends a REQUEST message to all other processes to request their permission to enter the critical section. A process sends a REPLY message to a process to give its permission to that process

7. List the conditions of Maekawa's algorithm.(U)

rithm. The request sets for sites (i.e., quorums) in Maekawa's algorithm are constructed to satisfy the following conditions:

M1 $(\forall i\, \forall j : i \neq j, 1 \leq i, j \leq N :: R_i \cap R_j \neq \phi)$.
M2 $(\forall i : 1 \leq i \leq N :: S_i \in R_i)$.
M3 $(\forall i : 1 \leq i \leq N :: |R_i| = K)$.
M4 Any site S_j is contained in K number of R_is, $1 \leq i, j \leq N$.

8. List the two design issues of Suzuki-Kasami's broadcast algorithm.(U)

- How to distinguishing an outdated REQUEST message from a current REQUEST message.
- How to determine which site has an outstanding request for the CS

9. **What is wait for graph(WFG) (R)**

The state of the system can be modeled by directed graph, called a wait-for graph (WFG). In a WFG, nodes are processes and there is a directed edge from node P1 to mode P2 if P1 is blocked and is waiting for P2 to release some resource. A system is deadlocked if and only if there exists a directed cycle or knot in the WFG.

10. Define Edge chasing.(R)

A distributed approach to deadlock detection uses a technique called edge chasing or path pushing. In this approach, the global wait-for graph is not constructed, but each of the servers involved has knowledge about some of its edges.

The servers attempt to find cycles by forwarding messages called probes, which follow the edges of the graph throughout the distributed system. A probe message consists of transaction wait-for relationships representing a path in the global wait-for graph.

11. Mention the three strategies for handling deadlock.(U)

- deadlock prevention
- deadlock avoidance
- deadlock detection

12. What are the issues in deadlock detection.(U)

- Detection of existing deadlocks
- Resolution of detected deadlocks.

13. **List various models of deadlocks**.(U)

- The single-resource model
- The AND model
- The OR model
- The AND-OR model
- The $\binom{P}{q}$ model
- Unrestricted model

14. Define single resource model.(R)

The single-resource model is the simplest resource model in a distributed system, where a process can have at most one outstanding request for only one unit of a resource.

15. Mention the four classes of distributed deadlock detection algorithms.(U)

- Path-pushing
- Edge-chasing
- Diffusion computation
- Global state detection.

16. Mention the features of Mitchell and Merritt's algorithm.(U)

- Only one process in a cycle detects the deadlock. This simplifies the deadlock resolution – this process can abort itself to resolve the deadlock. This algorithm can be improvised by including priorities, and the lowest priority process in a cycle detects deadlock and aborts.
- In this algorithm, a process that is detected in deadlock is aborted spontaneously,
  even though under this assumption phantom deadlocks cannot be excluded. It can be shown, however, that only genuine deadlocks will be detected in the absence of spontaneous aborts.

Part -A

1. What are the different models of deadlocks ? **April/May 2021**
2. What is the purpose of the wait-for-graph (WFG) ? Give an example for WFG. **April/May 2021**
3. What is the purpose of associating timestamp with events in Lamport's algorithm? **Nov./Dec. 2021**
4. Define deadlock. **Nov./Dec. 2021.**
5. What is deadlock? **Nov/Dec 2022**
6. Explain the term mutual exclusion. **Nov/Dec 2022**

Part -B

1. Discuss in detail the requirements that mutual exclusion algorithms should satisfy and discuss what metric we use to measure the performance of mutual exclusion algorithms. **April/May 2021**
2. List out the four classes of distributed deadlock detection algorithms and explain any two of them. **April/May 2021**
3. Outline Lamport's algorithm with an example. **Nov./Dec. 2021**
4. How we can achieve deadlock detection in distributed systems? Provide various models to carry out the same. **Nov./Dec. 2021**
5. External synchronization ensures internal synchronization. But the vice versa does not stand true. Justify. Explain Lamport's algorithm in brief. **April/May 2021**
6. Explain the Ricart-Agarwala algorithm with example. **Nov/Dec 2022**
7. Name and explain the different types of deadlock distributed system with the commonly used strategies to handle the deadlock with example. **Nov/Dec 2022**
8. Analyses the Suzuki-Kasami's broadcast algorithm for mutual exclusion in distributed system. **Nov/Dec 2022**
9. Discuss the metrics used to measure the performance of mutual exclusion.(U)
10. Explain Lamport's algorithm.(U)
11. Explain Ricart-Agarwala Algorithm.(U
12. Explain quorum based mutual exclusion Maekawa algorithm in detail.(U)
13. Explain the Suzuki-Kasami's broadcast algorithm.(U)
14. Explain Deadlock handling strategies and its issues.(U)
15. Discuss various models of deadlocks.(U)
16. Explain the classes of distributed deadlock detection algorithms.(U)
17. Explain Mitchell and Merritt's algorithm for the single-resource model.(U)
18. Explain Chandy–Misra–Haas algorithm for the AND model.(U)
19. Explain Chandy–Misra–Haas algorithm for the OR model.(U)
20. List out four classes of distributed deadlock detection algorithm(U)

UNIT IV RECOVERY & CONSENSUS

Checkpointing and rollback recovery: Introduction – Background and definitions – Issues in failure recovery – Checkpoint-based recovery – Log-based rollback recovery – Coordinated checkpointing algorithm – Algorithm for asynchronous checkpointing and recovery. Consensus and agreement algorithms: Problem definition – Overview of results – Agreement in a failure – free system – Agreement in synchronous systems with failures

PART – A

1. **Define checkpoint.(R)**
   A checkpoint can be saved on either the stable storage or the volatile storage depending on the failure scenarios to be tolerated.

2. **Define rollback recovery.(R)**
   Rollback recovery treats a distributed system application as a collection of processes that communicate over a network.

3. **Define rollback propagation.(R)**
   Upon a failure of one or more processes in a system, these dependencies may force some of the processes that did not fail to roll back, creating what is commonly called a rollback propagation.

4. **Define local checkpoint.(R)**
   A local checkpoint is a snapshot of the state of the process at a given instance and the event of recording the state of a process is called local check pointing.

5. **Mention the different types of messages.(U)**
   - In-transit messages
   - Lost messages
   - Delayed messages
   - Orphan messages
   - Duplicate messages

6. **List the categories of checkpoint based recovery**.(U)
   - Uncoordinated checkpointing
   - Coordinated checkpointing
   - Communication-induced checkpointing

7. **Differentiate blocking and non-blocking checkpoint coordination.(U)**

   Blocking: Blocking algorithms force all relevant processes in the system to block their underlying computation during checkpointing latency.

Non-blocking: In non-blocking algorithms applications processes are not blocked when checkpoints are being taken

8. Define modelbased checkpointing (R)

In model-based checkpointing,the system maintains checkpoints and communication structures that prevent the domino effect or achieve some even stronger properties.

9. Define indexbased checkpointing (R)

In index-based checkpointing, the system uses an indexing scheme for the local and forced checkpoints, such that the checkpoints of the same index at all processes form a consistent state.

10. Differentiate Deterministic and non-deterministic events?(U)

DETERMINISTIC ALGORITHM	NON-DETERMINISTIC ALGORITHM
For a particular input the computer will give always same output.	For a particular input the computer will give different output on different execution.
Can solve the problem in polynomial time.	Can't solve the problem in polynomial time.
Can determine the next step of execution.	Cannot determine the next step of execution due to more than one path the algorithm can take.

11. List the conditions of no-orphans consistency.(U)

Depend(e): the set of processes that are affected by a non-deterministic event e. This set consists of p, and any process whose state depends on the event e according to Lamport's happened before relation

• Log(e): the set of processes that have logged a copy of e's determinant in their volatile memory.

• Stable(e): a predicate that is true if e's determinant is logged on the stable storage.

12. Define optimistic and pessimistic logging.(R)

Pessimistic logging protocols assume that a failure can occur after any non-deterministic event in the computation. This assumption is "pessimistic" since in reality failures are rare.

In optimistic logging protocols, processes log determinants asynchronously to the stable storage. These protocols optimistically assume that logging will be complete before a failure occurs.

13. Define casual logging (R)

Causal logging combines the advantages of both pessimistic and optimistic logging at the expense of a more complex recovery protocol. Like optimistic logging, it does not require synchronous access to the stable storage except during output commit. Like pessimistic logging, it allows each process to commit output independently and never creates orphans, thus isolating processes from the effects of failures at other processes.

14. State the checkpointing algorithm.(U)

The checkpointing algorithm assumes that a single process invokes the algorithm at any time to take permanent checkpoints. The algorithm also assumes that no process fails during the execution of the algorithm.

15. State the byzantine agreement problem.(U)

The Byzantine agreement problem requires a designated process, called the source process, with an initial value, to reach agreement with the other processes about its initial value, subject to the following conditions:

• **Agreement** All non-faulty processes must agree on the same value.
• **Validity** If the source process is non-faulty, then the agreed upon value by all the non-faulty processes must be the same as the initial value of the source.
• **Termination** Each non-faulty process must eventually decide on a value.

16. State the consensus problem? (U)

The consensus problem differs from the Byzantine agreement problem in that each process has an initial value and all the correct processes must agree on a single value .

• **Agreement** All non-faulty processes must agree on the same (single) value.
• **Validity** If all the non-faulty processes have the same initial value, then the agreed upon value by all the non-faulty processes must be that same value.
• **Termination** Each non-faulty process must eventually decide on a value.

17. Drawbacks of Rollback and check point algorithm

1.Mentioned for model-based protocols
2. Depends on when checkpoint-decision

PART - A

1. What do you mean by local checkpoints ? **April/May 2021**
2. What is the drawback of a checkpoint based rollback recovery approach ? **April/May 2021**
3. List the benefits of recovery. **Nov./Dec.2021**
4. Why coordination is required in distributed systems? **Nov./Dec.2021**
5. State the use of rollback recovery. **Nov/Dec 2022**
7. What is consensus distributed system. **Nov/Dec 2022**

PART - B

1. Write about the issues in failure recovery and discuss about any two recovery mechanisms. **Nov./Dec.2021**
2. Describe the role of consensus and agreement algorithms in distributed, system along with its underlying structure , benefits and an example. **Nov./Dec.2021**
3. What are the key assumptions underlying while designing agreement algorithms and brief them? **April/May 2021**
4. Describe the issues involved in a failure recovery with the help of a distributed computation. **April/May 2021**
5. Illustrate the different types of failures in distributed system and explain how to prevent them. **Nov/Dec 2022**
6. Illustrate briefly the two kinds of checkpoint with checkpoint algorithm. **Nov/Dec 2022**
7. Comparison between Checkpoint Schemes.(U)
8. Explain various types of messages of processes.(U)
9. Illustrate various issues in failure recovery.(U)
10. Explain logbased rollback recovery.(U)
11. Explain Koo-Toueg coordinated check pointing algorithm.(U)
12. Explain the byzantine agreement problem.(U)
13. Explain consensus algorithm for crash failures.(U)
14. Explain consensus algorithms for byzantine failures.(U)
15. Explain agreement in synchronous message passing systems with failures.(U)

UNIT V P2P & DISTRIBUTED SHARED MEMORY

Peer-to-peer computing and overlay graphs: Introduction – Data indexing and overlays – Chord – Content addressable networks – Tapestry. Distributed shared memory: Abstraction and advantages – Memory consistency models –Shared memory Mutual Exclusion.

1. **What is data indexing? (U)**

Data Indexing allows the physical data independence from the applications. Indexing mechanisms can be classified as being centralized, local, or distributed.

2. **Define Centralized indexing. (R)**

Centralized indexing entails the use of one or a few central servers to store references (indexes) to the data on many peers. The DNS lookup as well as the lookup by some early P2P networks such as Napster used a central directory lookup.

3. **Define Distributed indexing. (R)**

Distributed indexing involves the indexes to the objects at various peers being scattered across other peers throughout the P2P network. In order to access the indexes, a structure is used in the P2P overlay to access the indexes.

4. **How does distributed Hash Table works? (AN)**

A typical DHT uses a flat key space to associate the mapping between network nodes and data objects/files/values. Specifically, the node address is mapped to a logical identifier in the key space using a consistent hash function.

5. **List out the types of Data indexing. (R)**

Indexing mechanisms can be classified as

- Centralized Indexing
- Distributed Indexing
- Local Indexing

6. **Define Peer-to-peer (P2P) Network. (R)**

Any node in a P2P network can act as a server to others and, at the same time, act as a client. Communication and exchange of information is performed directly between the participating peers and the relationships between the nodes in the network are equal.

7. **What is the drawback of DNS (domain name service) in distributed systems? (U)**

DNS (domain name service) is used to provide a lookup from host names (logical names) to IP addresses. Special DNS servers are required, and manual configuration of the routing information is necessary to allow requesting client nodes to navigate the DNS hierarchy. DNS is confined to locating hosts or services and host names need to be structured as per administrative boundary regulations.

8. Define churn. (R)

The ongoing entry and exit of various nodes, as well as dynamic insertion and deletion of objects is termed as churn.

9. List out the desirable characteristics and performance features of P2P systems. (R)

- Self-organizing
- Distributed control
- Role symmetry for nodes
- Anonymity
- Naming mechanism
- Security, authentication, trust

10. List out the performance features of P2P systems.(R)

- Large combined storage, CPU power, and resources
- Fast search for machines and data objects
- Scalable
- Selection of geographically close servers
- Efficient management of churn
- Redundancy in storage and paths

11. Define P2P overlay.(R)

P2P overlay is a logical graph among the peers that is used for the object search and object storage and management algorithms. P2P overlay is the application layer overlay, where communication between peers is point-to-point (representing a logical all-to-all connectivity) once a connection is established.

12. What are the steps involved in Chord distributed hash table? (U)

Two steps are involved:

1. Map the object/file/value to its key in the common address space.
2. Map the key to the node in its native address space using lookup. The design of lookup is the main challenge.

13. Explain simple object location algorithm in Chord(U)

(Variables)
Integer: successor ← initial value;
(1) i.Locate_Successor (key), where key ≠ i:
(1a) **if** key ∈ (i, successor) **then**
(1b) **return** (successor)
(1c) **else return** (successor.Locate_Successor(key)).

14. Define finger table.

Each node i maintains a routing table, called the finger table, with at most O(log n) distinct entries, such that the xth entry ($1 \leq x \leq m$) is the node identifier of the node succ($i+2^{x-1}$). This is denoted by
i.finger[x] = succ ($i+2^{x-1}$).

15. How does The Chord protocol works? (U)

The Chord protocol proposes that, rather than maintain a single successor, each node maintains a list of α successors, which are the node's first α successors. If the first successor does not respond, the node can try the next successor from the list, and so on. Only the simultaneous failure of all α the successors can then cause the protocol to fail.

16. Define content-addressible network (CAN). (R)

A content-addressible network (CAN) is essentially an indexing mechanism hat maps objects to their locations in the network.

17. List out the features of good CAN design. (R)

- Distributed
- Fault-Tolerant
- Scalable
- Independent of The Naming Structure
- Implementable at The Application Layer
- Autonomic.

18. Write down the basic operations supported by CAN.(R)

A CAN supports three basic operations:

- Insertion
- Search
- Deletion

19. What are the three core components of a CAN design? (U)

The three core components of a CAN design are the following:

1. Setting up the CAN virtual coordinate space, and partitioning it among the nodes as they join the CAN.
2. Routing in the virtual coordinate space to locate the node that is assigned the region containing $\vec{p}$.
3. Maintaining the CAN due to node departures and failures.

20. List down the design techniques that improve the performance factors in CAD Design. (R)

The following design techniques aim to improve one or more of the performance factors:

- Per-Hop Latency
- Path Length
- Fault Tolerance
- Availability
- Load Balancing

21. Write down the performance factors in CAD Design. (R)

- Multiple dimensions
- Multiple realities
- Delay latency
- Overloading coordinate regions
- Multiple hash functions
- Topologically sensitive overlay

22. Define surrogate root. (R)

If there exists a node v such that vid $=O_{GR}$, then v is the root of identifier O_G. If such a node does not exist, then a globally known deterministic rule is used to identify another unique node sharing the largest common prefix with O_G that acts as the surrogate root.

23. Determine the time complexity to search an object in Tapestry System. (AN)

The time complexity to search an object in Tapestry System is expected to take logb2mhops.

24. What are the advantages of the Distributed shared memory? (U)

- Communication across the network is achieved by the read/write abstraction that simplifies the task of programmers.
- A single address space is provided.
- If a block of data needs to be moved, the system can exploit locality of reference to reduce the communication overhead.
- DSM is often cheaper than using dedicated multiprocessor systems, because it uses simpler software interfaces and off-the-shelf hardware.
- There is no bottleneck presented by a single memory access bus.
- DSM effectively provides a large (virtual) main memory.
- DSM provides portability of programs written using DSM.

25. What is Distributed shared memory? (U)

Distributed shared memory (DSM) is an abstraction provided to the programmer of a distributed system. It gives the impression of a single monolithic memory, as in traditional von Neumann architecture. Programmers access the data across the network using only read and write primitives.

26. What are the main issues in designing a DSM system? (U)

The main issues in designing a DSM system are the following:

- Determining what semantics to allow for concurrent access to shared objects.
- Determining the best way to implement the semantics of concurrent access to shared data.
- Selecting the locations for replication (if full replication is not used), to optimize efficiency from the system's viewpoint.
- Determining the location of remote data that the application needs to access, if full replication is not used.
- Reducing communication delays and the number of messages that are involved under the covers while implementing the semantics of concurrent access to shared data.

27. Define Memory coherence. (R)

Memory coherence is the ability of the system to execute memory operations correctly.

28. Explain the causality relation for shared memory systems.(U)

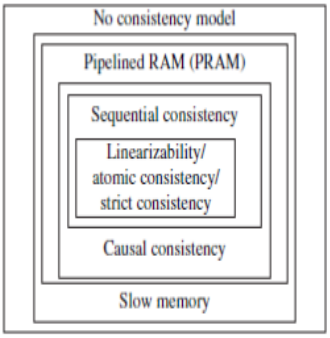
The causality relation for shared memory systems is defined as follows:

- **Local order-** At a processor, the serial order of the events defines the local causal order.
- **Inter-process order-** A Write operation causally precedes a Read operation issued by another processor if the Read returns a value written by the Write.
- **Transitive closure -**The transitive closure of the above two relations defines the (global) causal order.

29. Define slow memory model. (R)

Slow memory model represents a location-relative weakening of the PRAM model. In this model,only all Write operations issued by the same processor and to the same memory location must be observed in the same order by all the processors.

30. Draw a strict hierarchy of the memory consistency models.(U)



31. List out the properties of Weak consistency model.(R)

Weak consistency has the following three properties which guarantee that memory is consistent at the synchronization points:

- Accesses to synchronization variables are sequentially consistent.
- No access to a synchronization variable is allowed to be performed until all previous writes have completed everywhere.
- No data access (either Read or Write) is allowed to be performed until all previous accesses to synchronization variables have been performed.

32. What is the drawback of weak consistency model? (U)

The drawback of weak consistency is that when a synchronization variable is accessed, the memory does not know whether this is being done because the process is finished writing the shared variables (exiting the CS) or about to begin reading them (entering the CS).

33. What are the requirements needed for the critical section problem? (U)

- Mutual Exclusion
- Bounded Waiting
- Progress

34. Difference between shared memory and distributed memory? (U)

SHARED MEMORY	DISTRIBUTED MEMORY
• Shared memory paradigm gives the systems illusion of physically shared memory • Exist only virtually	• Distributed memory paradigm process with shared address space • DM provides a virtual address space shared among processes on loosely coupled processors

PART A

1. List out the characteristics of P2P systems. **April/May 2021**
2. What is the difference between shared memory and distributed memory? **April/May 2021**
3. Outline the benefits of peer-to-peer systems. **Nov./Dec.2021**
4. What is the need for memory consistency models? **Nov./Dec.2021**
5. What do you understand by two lines peer-peer computing? **Nov/Dec 2022**
6. Define: Data indexing **Nov/Dec 2022**

PART B

1. What do you understand about Content-Addressable Networks (CAN) ? Explain how it is useful in P2P networks. **April/May 2021**
2. Describe in detail about Distributed Shared Memory (DSM) and its application. **April/May 2021**
3. Elaborate the role of data indexing and overlays with an example. **Nov./Dec.2021**
4. Detail the structure of distributed shared memory along with real time challenges in implementing the same. **Nov./Dec.2021**
5. Explain the different types of Overlay network with its advantages and disadvantages. **Nov/Dec 2022**
6. Critically examine the different types of distributed shared memory with its advantages. **Nov/Dec 2022**
7. Discuss in detail about Napster and Application Overlays.(U)
8. Explain in detail about Data Indexing and its types.(U)
9. Describe about Chord distributed hash table with example.(U)
10. Write short notes on the following(R)
    i. Scalable lookup
    ii. Simple lookup
11. Discuss in detail about Managing Churn Algorithm and its complexity.(U)
12. Explain how initialization, routing and maintenance are performed in Content Addressible Networks (CAN).(AN)
13. Write short notes on the following (R)
    i. CAN optimizations
    ii. CAN complexity
14. Describe about routing and overlays in Tapestry System.(U)
15. Analyze how node insertion and deletion is performed in Tapestry System. (AN)
16. Write short notes on the following(R)
    i. CAN optimizations
    ii. CAN complexity
17. Explain in detail about distributed shared memory and its application.(U)
18. Describe in detail about Lamport's bakery algorithm and fast mutual exclusion.(U)
19. Explain about Hardware support for mutual exclusion.(U)
20. Discuss about Memory consistency models.(U)